



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Group Art Unit: 2124
Examiner: William H. Wood

IFW
AF 2124
\$B

In re Application of: Thomas D. Hartnett et al.
Title: Pipeline Controller for Providing Independent Execution between the Preliminary and Advanced Stages of a Synchronous Pipeline
Serial No.: 09/468,051
Filed: December 20, 1999
Docket No.: RA 5271K
Customer No. 27516

Date: July 16, 2004

Commissioner of Patents
MS Appeal Brief - Patents
P O Box 1450
Alexandria, VA 22313-1450

APPELLANT'S BRIEF TRANSMITTAL

Sir:

Transmitted herewith is an Appellant's Brief for this application. Applicant is other than a small entity.

We are transmitting herewith the attached:

- Appellant's Brief Filed Under 37 C.F.R. 1.193 (d) in Triplicate.

FEE PAYMENT

Please charge Account No. 19-3790 the sum of \$330.00 filing fee. If any additional fee is required, charge Account No. 19-3790.

A duplicate of this transmittal is attached.

Respectfully submitted,

Beth L. McMahon

Beth L. McMahon
Reg. No.: 41,987
Tel. No.: (651) 635-7893
Unisys Corporation
M.S. 4773
P.O. Box 64942
St. Paul, MN 55164-0942

I hereby certify that this correspondence is being deposited in the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450 on July 16, 2004.

Beth L. McMahon
Attorney for Applicants

Beth L. McMahon
Signature

July 16, 2004
Date of Signature



APPEAL TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Group Art Unit: 2183
Examiner: William H. Wood

July 16, 2004

Customer Assignment No. 027516
Serial No.: 09/468,051
Filed: December 20, 1999
In re Application of: Thomas D. Hartnett, John S. Kuslak, and Gary J. Lucas
Title: PIPELINE CONTROLLER FOR PROVIDING INDEPENDENT
EXECUTION BETWEEN THE PRELIMINARY AND ADVANCED
STAGES OF A SYNCHRONOUS PIPELINE
Docket No.: RA-5271K


APPELLANT'S BRIEF

MS Appeal Brief - Patents
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

This brief is being submitted within two (3) months of receipt of the Notice of Appeal by the U.S. Patent and Trademark Office. This brief, which is filed in triplicate, is transmitted with the required fee. A petition for a **ONE-MONTH EXTENSION** of time under 37 CFR 1.136(a) is attached herewith. Permission is hereby granted to charge deposit account number 19-3790 for any errors in fee calculation. Appellants request this Appeal Brief be made of record and fully considered.

CERTIFICATE OF MAILING (37 CFR 1.8(a))

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Box MS Appeal Brief - Patents, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below:


(Beth McMahon)

July 16, 2004
(Date)

07/20/2004 LWONDIM1 00000007 193790 09468051

01 FC:1402 330.00 DA

TABLE OF CONTENTS

Real Party in Interest.....	3
Related Pending Appeals or Interferences.....	3
Status of Claims.....	3
Status of the Amendments.....	3
Summary of the Invention.....	4
Issues.....	10
Grouping of the Claims.....	11
Arguments.....	11
Conclusion and Request for Relief.....	28
Appendix A.....	30

Real Party In Interest

The real party in interest is Unisys Corporation, with an address as follows:

Unisys Corporation
Township Line and Union Meeting Roads
Blue Bell, Pennsylvania 19424

Unisys Corporation is the real party in interest through an assignment from all of the inventors of their entire interest, recorded on 12/20/1999 in the USPTO at Reel/Frame 010650/0869.

Related Pending Appeals or Interferences

There are no pending appeals or interferences related to the subject Appeal.

Status of Claims

Claims 1-7 and 21-46 remain pending, and stand finally rejected.

Status of the Amendments

A first amendment was submitted on October 25, 2002 to amend Claims 1, 7, 8 and 14 of originally presented Claims 1-20. This amendment has been entered.

A second amendment filed April 30, 2003 accompanied a Request for Continuing Examination (RCE), and amended Claim 1, canceled Claims 8-20, and added Claims 21-46. This amendment has been entered.

No further amendments have been submitted or entered.

A clean copy of Claims 1-7 and 21-46, as amended, is provided as Appendix A.

Summary of the Invention

Applicants' invention provides an improved synchronous instruction pipeline system and method for a data processing system. This instruction pipeline provides fetch and execution stages that are independently operable, yet directly interconnected.

Applicants' system can best be understood by providing a general discussion of pipeline architectures. As is known in the art, an instruction pipeline includes multiple logic sections, or "stages", each of which performs one of the operations required to process an instruction to completion. After an instruction enters the pipeline, it is processed in an "assembly line" fashion, moving from one stage to the next until instruction execution is completed. At each stage, a task such as instruction decode, operand address generation, or the fetching of one or more operands, is initiated.

According to the current invention, an instruction pipeline is provided that includes a first number of fetch stages and a second number of execution stages. In an exemplary embodiment, the fetch stages include stages 0Y

through 3Y, and the execution stages comprise stages 1X through 6X.¹ The fetch stages control the operations that must occur before signals are provided to the execution logic of the instruction processor. These operations include the fetching and decoding of the instruction. After an instruction has been processed by each of the fetch stages, it is provided to the execution stages, which control operations such as operand address generation, operand retrieval, arithmetic processing, and the storing of any results.²

When Applicants' instruction pipeline is operating in a fully overlapped manner, each of the fetch and execution stages is processing a different instruction at a given time. In this case, the instructions move from one pipeline stage to the next in lock-step with one another. Because the pipeline is synchronous, this movement occurs during each cycle of the system clock.

Some situations can disrupt the type of fully overlapped processing discussed above. For instance, sometimes an instruction may "stall" within an execution stage of the pipeline so that the instruction is no longer advancing. As an example, Applicants' pipeline is capable of executing "extended-mode" instructions that require more than one clock cycle to complete the first of the execution stages. When an extended-mode instruction resides within the first execution stage, subsequent execution stages become empty because instructions have ceased to advance into these stages.³ As another example, if an attempt to fetch an operand from a cache memory results in a cache miss,

¹ Applicants' Specification ("Specification"), page 12 line 5 - page 13 line 3 in reference to Figure 1.

² Applicants' Specification ("Specification"), page 6 lines 11-17.

instruction execution is delayed while the operand is retrieved from a slower memory. This, likewise, causes one or more downstream execution stages to empty.⁴

In a manner similar to that discussed above with respect to the execution stages, one or more of the fetch stages may also become empty. For example, a cache miss may occur during the retrieval of an instruction. This will cause a delay in instruction availability so that some of the fetch stages do not contain an instruction. This may also occur when a flush operation must be initiated to clear a sequence of instructions from the fetch stages of the pipeline after a branch misprediction occurs.⁵

To optimize operation of the data processing system during the types of situations discussed above, Applicants' pipeline decouples the fetch stages from the execution stages. This allows instructions to enter into, and advance between, the fetch stages of the pipeline regardless of whether instructions are advancing between the execution stages.⁶ This may be illustrated by the following example. Consider a situation wherein a cache miss occurs while one of the fetch stages is retrieving an instruction. While waiting for this instruction to be returned from a slower memory, instructions stored within the downstream pipeline stages continue to advance within the pipeline. This causes some of the downstream fetch stages to become empty. During this time, an extended-mode instruction enters the first execution stage. As previously discussed, this

³ Specification page 6 line 26 through page 7 line 7.

⁴ Specification page 8 lines 15-21.

⁵ Specification page 9 lines 9-16.

type of instruction will reside within the first execution stage for multiple clock cycles. This will prevent subsequent instructions in the instruction stream from entering the execution stages of the pipeline until the first stage of execution has completed for the extended-mode instruction.

In a situation such as that described above, Applicants' pipeline optimizes throughput by decoupling the fetch and execution stages. According to the invention, instruction fetching is allowed to continue even though no additional instructions are entering the execution stages. Thus, in the current example, when the instruction that resulted in the cache miss finally becomes available from memory, that instruction may enter, and advance between, the various fetch stages. All decode operations may be completed for this instruction so that it is available to enter the first of the execution stages after processing has completed for the extended-mode instruction⁷. Moreover, additional instructions may be retrieved from memory while the extended-mode instruction completes the first execution stage.⁸ These instructions will fill any of the fetch stages that have emptied because of the cache miss situation.

As can be appreciated from the foregoing discussion, the decoupling of the fetch and execution stages increases the chances that an instruction will be available for entry into the first stage of execution when a previous instruction has completed this stage. It further ensures that overlapped execution will resume as quickly as possible after a stall has occurred in an execution stage or

⁶ Specification page 7 lines 8-11.

⁷ Specification page 8 lines 10-12.

⁸ Specification page 8 lines 5-8.

a fetch stage has become empty. If the fetch and execution stages are not decoupled in this manner, the stalling of an execution stage will cause the fetch stages to likewise stall so that empty fetch stages remain empty. When instructions are once again allowed to advance within the execution stages, some fetch stages remain empty so that the system is not operating in a fully overlapped manner. Therefore, if fetch and execution stages remain decoupled, processing throughput is diminished.

As discussed above, Applicants' system and method provides fetch and execution stages that are decoupled to allow instructions to advance within the fetch stages independently of the times at which they advance within the execution stages. Applicants' invention provides this capability while also allowing the fetch stages to be directly interconnected to the execution stages. This means that the last fetch stage, shown as stage 3Y in Applicants' Figures, is interconnected to provide instructions directly to the 1X execution stage.⁹ This allows instructions that have completed the last fetch stage to begin execution within the first execution stage without incurring any unnecessary delay.

One alternative approach to Applicants' system for directly interconnecting fetch and execution stages involves *indirectly* connecting these stages via a storage device. For example, a queue may be inserted between the last fetch stage and the first execution stage. According to this method, instructions that are awaiting entry into the first execution stage are stored into

⁹ See, for example, Figure 11, which illustrates the decode logic 260 of stage 3Y coupled to provide instructions on line 122 to the instruction decode dispatch logic 124 of Figure 7, which performs the 1X execution stage.

the queue by the last fetch stage. When the first execution stage becomes available to receive another instruction, it retrieves the next instruction from this queue. While this alternative mechanism provides a simple approach to decoupling the fetch and execution stages, it imposes delay within the pipeline. This is particularly true in cases wherein the time to store, then retrieve, an instruction cannot be made transparent to the pipeline.

Rather than utilize a storage device in the manner discussed above to decouple pipeline stages, Applicants' system provides a control circuit to perform this task.¹⁰ This circuit, which is illustrated in Applicants' Figures 10 and 11, allows instructions to advance between fetch stages of a pipeline even if an instruction is stalled within the execution stages. At the same time, this circuit allows instructions to be provided directly from the last fetch stage to the first execution stage so that no unnecessary delays are imposed upon the pipeline.

Aspects of Applicants' invention discussed above are summarized in Applicants' representative Claim 1, which describes:

"...a synchronous instruction pipeline, comprising:

a pipeline execution circuit to process a first predetermined number of instructions...each... being in a respectively different stage of execution...; and

a pipeline fetch circuit coupled to provide each of the first predetermined number of instructions directly to said pipeline execution circuit, the pipeline fetch circuit to retain a second predetermined number of instructions... each ... being in a respectively different stage of processing within said pipeline fetch circuit, an instruction being capable of advancing to a next stage of execution within said pipeline fetch

¹⁰ See Applicants' Figures 10 and 11.

circuit...independently of the times at which instructions advance to a next stage of execution within *said pipeline execution circuit*."¹¹

Issues

I. Whether Claims 1, 21, 27, 32, 39, 40 and 46 are unpatentable under 35 USC §103(a) over U.S. Patent Number 6,112, 295 to Bhamidipati et al. (hereinafter "Bhamidipati") in view of Hayes, John P., "Computer Architecture and Organization", (hereinafter "Hayes").

This issue may be divided into the following sub-issues A.) - C.):

A.) Whether the Bhamidipati elements cited by the Examiner render Claims 1, 21, 27, 32, 39, 40 and 46 unpatentable in view of Hayes.

B.) Whether any other elements of Bhamidipati render Claims 1, 21, 27, 32, 39, 40 and 46 unpatentable in view of Hayes.

C.) Whether the Examiner has set forth a prima facie case for obviousness with respect to Claims 1, 21, 27, 32, 39, 40 and 46.

II. Whether Claims 2-6, 22-25, 28-30, 33-38, 41-43, and 45 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes, and further in view of U.S. Patent Number 6,026,477 to Kyker et al. (hereinafter, "Kyker").

III. Whether Claim 7 is unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes and Kyker, and further in view of U.S. Patent Number 5,577,259 to Alferness et al. (hereinafter, "Alferness").

¹¹ Claim 1 lines 3-6, and 9-17, emphasis added.

IV. Whether Claims 26, 31, and 44 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes and further in view of Alferness.

Grouping of the Claims

Claims 1-6, 21-25, 27-30, 32-43, and 45-46 stand or fall together; and

Claims 7, 26, 31, and 44 stand or fall together.

Arguments

I. Whether Claims 1, 21, 27, 32, 39, 40 and 46 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes.

Before considering this issue in detail, the Bhamidipati system is summarized for discussion purposes. Bhamidipati discloses a pipeline system having multiple stages. A decoupling queue is provided to decouple a pipeline stage N from a next stage N+1.¹² The decoupling queue operates so that during a given clock cycle, pipeline stage N may store an instruction to a location within the decoupling queue, and pipeline stage N+1 may retrieve an instruction from the same, or a different, location within this queue.¹³ A write pointer is used to point to the storage location that will receive the next stored instruction. Similarly, a read pointer is used to point to the storage location from which the next instruction will be retrieved.¹⁴

¹² Bhamidipati Abstract lines 5-8.

¹³ Bhamidipati column 3 lines 20-23.

¹⁴ Bhamidipati column 4 lines 12-14.

The Bhamidipati decoupling queue can operate in one of two modes. In a Write-Read (WR) mode, stage N stores an instruction into the decoupling queue before stage N+1 reads an instruction.¹⁵ In contrast, when operating in Read-Write (RW) mode, stage N+1 first retrieves an instruction from the decoupling queue before stage N stores another instruction to the queue. The timing for these operations is illustrated by Bhamidipati Figure 9. Specifically, in WR mode, writing of an instruction occurs during period 902 followed by instruction retrieval during period 904. In RW mode, the opposite occurs, with instruction retrieval being performed during duration 902 followed by the storing of an instruction during period 904.¹⁶

With the foregoing summary of the Bhamidipati system available for discussion purposes, the three sub-issues set forth above within respect to Issue I are considered in turn.

A.) Whether the Bhamidipati elements cited by the Examiner render Claims 1, 21, 27, 32, 39, 40 and 46 unpatentable in view of Hayes.

The Examiner's assertions concerning the Bhamidipati are best considered in reference to Bhamidipati Figure 3. That Figure illustrates a circuit 502 that includes several pipeline stages 300, 302, 304, and 308. The Examiner cites these stages as disclosing Applicants' fetch stages.¹⁷ The Examiner further asserts that Applicants' execution stages are disclosed by the

¹⁵ Bhamidipati column 4 lines 7-10 and 18-24.

¹⁶ Bhamidipati column 5 lines 25-50.

¹⁷ Paper Number 12, page 3, section 5, lines 11-14.

execution element 106 of Figure 1.¹⁸ Elements 300, 302, and 304 correspond to pipe stage F 100, of Figure 1, and instruction decode unit 308 of Figure 3 correlates to decode stage 102 of Figure 1.¹⁹ Stage 104 may also be considered a fetch stage, since it precedes stage 106 cited by the Examiner as teaching Applicants' execution stages.

Next, the supposed teachings of Bhamidipati are considered in reference to the language of Applicants' representative Claim 1. According to this Claim, instructions are capable of advancing to a next stage within the pipeline fetch circuit independently of the times at which instructions advance to a next stage of execution within the execution circuit. This type of independent operation is not disclosed by the Bhamidipati fetch and execution stages, as cited by the Examiner. Specifically, the Bhamidipati stages 308 and 104 appear to operate in lock-step with execution stage 106. That is, once an instruction has been retrieved from decoupling queue 306 for processing within stage 308 of Figure 3, it appears that the way in which the instruction will advance will be entirely determined by the way instructions advance within the subsequent stages. There is no indication whatsoever in Bhamidipati that an instruction will advance from stage 308 to stage 104 while an instruction is stalled in stage 106.

To reiterate, according to the Examiner's analysis, the Bhamidipati decoupling queue 306 decouples two fetch stages 304 and 308. This use of a decoupling queue *between* two fetch stages does not provide a circuit that allows instructions to advance within the fetch stages independently of when

¹⁸ Paper Number 12, page 3, section 5, lines 7-10

instructions advance within the execution stage, as is described in Applicants' Claims. As previously discussed, this is because once an instruction enters fetch stage 308, any further advancement of this instruction appears to be directly controlled by whether instructions are advancing within the execution stage 106, and no indication of Bhamidipati is provided to the contrary.

The Examiner's interpretation of Bhamidipati as discussed above does not teach or suggest Applicants' system and method for yet another reason. In Applicants' Claim 1, each instruction retained within the fetch circuit at a given time is in a respectively different stage of processing.²⁰ As defined in Applicants' Specification, a stage is a functional operation that is performed by a respective logic section of the IP.²¹ Examples of such functional operations include the decoding of an instruction, generating an operand address, fetching an operand, performing manipulation on the operands, and etc.²² This use of the term "stage" is consistent with the way in which the term is used in the art in general, and in Bhamidipati. For example, Bhamidipati describes each pipe stage as "...complet[ing] a part of an instruction".²³

As discussed above, the Examiner states that Applicants' pipeline fetch circuit is taught by Bhamidipati stages 300, 302, 304, and 308.²⁴ Since the decoupling queue 306 interconnects stages 304 and 308 and affects the way these stages operate, the decoupling queue must be part of the Bhamidipati

¹⁹ Bhamidipati column 3 lines 44-48.

²⁰ Claim 1 lines 10-13.

²¹ Applicants' Specification page 12 lines 8-12.

²² Specification page 12 lines 11-22.

²³ Bhamidipati column 1 lines 16-17.

circuit 502 that supposedly teaching Applicants' fetch circuit. However, as discussed above, the decoupling queue is not a "stage" as that term is defined in Applicants' Specification and used in the art. Therefore, some of the instructions in the Bhamidipati circuit 502 are not in any "stage" at all, but are rather just stored within the decoupling queue. Thus, Bhamidipati does not teach Applicants' fetch circuit wherein each of the instructions retained by the fetch circuit are in a respectively different stage of processing. For this additional reason, Bhamidipati does not teach or suggest Applicants' claimed invention.

Next, the teaching of Hayes is considered. Hayes is a reference that pre-dates the Bhamidipati patent by more than two decades.²⁵ The Hayes reference discusses relatively primitive instruction pipelines that include a fetch stage coupled to an execution stage.²⁶ Hayes provides no teaching or suggestion whatsoever on allowing instructions to advance within fetch stages independently of how they advance within the execution stages. Hayes most certainly does not teach the aspects of Applicants' representative Claim 1 wherein the fetch and execution circuits are directly coupled yet independently operable with respect to the advancing of instructions.

Turning to the Examiner's analysis of Hayes, the Examiner cites the Hayes reference as teaching the concept that fetch and execution stages may

²⁴ Paper 12 page 3, last full paragraph.

²⁵ Copyright 1978 by McGraw-Hill, Inc.

²⁶ Hayes page 223, last full paragraph.

be directly coupled.²⁷ This assertion is not understood for several reasons. First, according to the Examiner's analysis of Bhamidipati that is discussed above, the Bhamidipati fetch stages are directly interconnected to the Bhamidipati execution stages. Thus, the need to cite Hayes is not understood.

In addition to the foregoing, Hayes discusses a 1970s pipeline architecture that is an early precursor to the system described in Bhamidipati. In fact, the Bhamidipati background section begins with a discussion of a Hayes-type pipeline system wherein "[t]he pipe stages are connected one to the next to form a pipe, where instructions enter at one end, are processed through the stages, and exit at the other end."²⁸ The Bhamidipati description continues with the evolution of pipeline architectures, stating that "...in order to ensure the optimal performance of a pipeline, one method involves inserting queues in the pipeline to decouple these pipe stages."²⁹ This method is said to allow stages to carry on tasks independently. However, Bhamidipati also notes that this method imposes "...delay to perform either a read or write operation..." which can be "prohibitively lengthy in view of a diminishing clock cycle."³⁰ For this reason, Bhamidipati describes an improvement to this decoupling queue mechanism that seeks to shorten this delay by using a queue supporting both read and write operations during a single clock cycle.³¹

²⁷ Paper 12 page 5 line 2-3.

²⁸ Bhamidipati column 1 line 17-19.

²⁹ Bhamidipati column 1 lines 24-26.

³⁰ Bhamidipati column 1 lines 41-43.

³¹ Bhamidipati column 1 lines 56-61.

To summarize, the Bhamidipati background section commences with a description of the Hayes 1970s pipeline architecture, and traces improvements that culminate in the addition of the Bhamidipati decoupling queue. Therefore, it is Bhamidipati that is building upon the Hayes teachings to arrive at the decoupling queue mechanism rather than Hayes providing any teaching that could add to Bhamidipati. One skilled in the art would most certainly not be motivated to modify Bhamidipati with teachings of Hayes, and the cited combination of references is therefore improper, as will be discussed further below.

Even assuming one skilled in the art were motivated somehow to add some teaching of Hayes to the Bhamidipati system, it is not understood how this supposed teaching would motivate one skilled in the art to arrive at Applicants' claimed invention. Hayes provides no teaching whatsoever on how to obtain a pipeline wherein the fetch and execution circuits are directly coupled yet are also independently operable in the manner claimed by the Applicants. Neither Bhamidipati or Hayes, alone or in combination, teaches or suggests this type of system, and the Bhamidipati elements cited by the Examiner do not render Claims 1, 21, 27, 32, 40 and 46 unpatentable in view of Hayes.

B.) Whether any other elements of Bhamidipati render Claims 1, 21, 27, 32, 39, 40 and 46 unpatentable in view of Hayes.

The foregoing analysis focuses on the Bhamidipati elements and embodiments that are cited by the Examiner in Paper 12 as teaching Applicants'

claimed invention. However, Bhamidipati discusses alternative embodiments wherein decoupling queues may be inserted between other pipeline stages.³² These additional embodiments are considered for completeness.

According to one alternative embodiment, a decoupling queue may be inserted between the Bhamidipati fetch and execution stages rather than between two of the fetch stages, as was considered above. This results in a situation wherein the last fetch stage 104 provides an instruction to be stored within the decoupling queue 306. Sometime later, the first execution stage retrieves this instruction for execution. This embodiment does not teach or suggest Applicants' system for providing instructions directly from the fetch stages to the execution stages, as claimed by representative Claim 1 and the remaining Claims.³³

The advantages provided by Applicants' claimed invention as compared to that of the Bhamidipati system become apparent when considering Bhamidipati Figure 9. Assume that the Bhamidipati decoupling queue 306 is empty such that the system is operating in read-write mode³⁴. In this mode, an instruction is stored to the decoupling queue 306 during time period 902. The instruction is subsequently retrieved from the decoupling queue by the first execution stage during time period 904³⁵. This instruction is therefore not available for processing by the first execution stage until time 908. Thus, in the situation wherein the queue is empty and the execution stage is waiting to

³² Bhamidipati column 3 lines 54 - 64.

³³ Claim 1 line 10.

³⁴ Bhamidipati column 4 lines 25-31.

receive another instruction from the last fetch stage, virtually an entire clock cycle is wasted because the decoupling queue is inserted between the fetch and execution stages. As previously discussed, a similar, more accentuated, problem is recognized by the Bhamidipati discussion of the prior art, as follows:

“As a processor further splits up its pipe stages and increases its clock speed, the duration of the decoupling queue’s setup time and its delay to perform either a read or a write operation can become prohibitively lengthy in view of a diminishing clock cycle. When such overhead equals to [sic] the processor clock cycle, no further pipelining is useful in enhancing a processor’s performance”³⁶

Although the Bhamidipati system succeeds in minimizing this discussed problem by providing a de-coupling queue that is capable of both storing and retrieving an instruction within the same clock cycle, the problem is not entirely eliminated. In those situations wherein the decoupling queue has become empty, a delay of more than half a clock cycle occurs between the time an instruction is available from the last fetch stage until the time it is received by the first execution stage. This delay is not insignificant in high-speed pipeline architectures.

Applicants’ system of representative Claim 1 addresses this type of problem by providing instructions directly from the fetch circuit to the execution circuit without utilization of a storage device such as queue.³⁷ This direct coupling is provided by the 3Y1 register 250 of Applicants’ Figure 11, which implements the last fetch stage, and which is directly interconnected to provide

³⁵ Bhamidipati column 4 lines 25-38.

³⁶ Bhamidipati column 1 lines 39-45.

an instruction to the first execution stage embodied by instruction decode dispatch logic 124 of Figure 7. In this manner, an instruction may be provided without delay from Applicants' last fetch stage to the first execution stage. Instead of employing a storage device to decouple these two stages, the control circuit shown in Applicants' Figures 10 and 11 is employed to provide the decoupling function.³⁸

The alternative Bhamidipati embodiment discussed in the above paragraphs provides a system wherein the fetch and execution stages are not directly coupled. Instead the fetch and execution stages are indirectly coupled through a decoupling queue. Next, it may be determined whether the teaching of directly coupling fetch and execution stages may be provided by Hayes.

As previously discussed, Hayes provides an early pipeline architecture from the 1970s. Bhamidipati expressly discusses how and why this type of early Hayes pipeline architecture should be modified to obtain the system of Bhamidipati. One skilled in the art would not be motivated by Hayes to modify Bhamidipati when Bhamidipati is teaching how to modify Hayes. Moreover, nothing in Hayes would motivate one skilled in the art to modify the Bhamidipati pipeline in any manner that would teach or suggest Applicants' pipeline system wherein fetch and execution stages are both directly interconnected and independently operable. Thus, the alternative embodiment discussed above does not render the Claims unpatentable in view of Hayes.

³⁷ Claim 1 line 10.

³⁸ See Specification pages 21-25 for a discussion of these Figures.

Finally, yet another characterization of the Bhamidipati system must be considered for completeness sake. In Paper 15, the Examiner states that the decoupling queue of Bhamidipati might be characterized as part of the Bhamidipati execution or fetch circuit rather than as a separate element that links these types of stages.³⁹ For example, a decoupling queue could be inserted before the execution stages 106 of the pipeline circuit of Figure 1. The decoupling queue and execution stages 106 may then be considered an “execution circuit” that is directly coupled to a fetch circuit, and which is then said to teach Applicants’ claimed invention.

The foregoing characterization does not teach or suggest Applicants’ claimed invention for reasons that are similar to that described in Section I above. Assume, for example, that the decoupling queue is included within the Bhamidipati execution circuit as described in the previous paragraph. This decoupling queue stores one or more instructions that are awaiting retrieval by the first execution stage. This queue does not perform any type of functional operations on the instructions. Therefore, this decoupling queue cannot be categorized as a “stage” of execution as that term is used in Applicants’ Specification and in the relevant arts because the decoupling queue is not performing any functional operations on the instruction.

Because the Bhamidipati decoupling queue is not a “stage of execution” as that term is used either in Applicants’ Specification or in Bhamidipati, the Examiner’s alternative characterization of Bhamidipati does not teach or

³⁹ Paper Number 15.

suggest Applicants' claimed invention. Applicants' representative Claim 1 describes an execution circuit wherein each instruction in the execution circuit is in a respectively different stage of execution. This is not true in the Bhamidipati "execution circuit" of the current analysis, wherein at least one instruction within this circuit will not be in a stage of execution at all, but instead will be stored within the decoupling queue. Viewed a different way, if the decoupling queue is considered part of the first execution stage, two instructions within the execution circuit will be in this same first execution stage.

To summarize, if the Bhamidipati decoupling queue is considered part of the execution circuit, all instructions within this execution circuit will not be within a different stage of execution as claimed by Applicants' Claims, and Bhamidipati does not teach or suggest Applicants' claimed invention. As discussed above in Section I, a similar analysis applies if the decoupling queue is instead considered part of a Bhamidipati "fetch circuit".

For all of the foregoing reasons, none of the embodiments, or alternative characterizations of Bhamidipati, render Applicants' Claims 1, 21, 27, 32, 39, 40, and 46 unpatentable in view of Hayes.

C.) Whether the Examiner has set forth a prima facie case for obviousness with respect to Claims 1, 21, 27, 32, 39, 40 and 46.

As discussed previously, the 1970-version pipeline illustrated in Hayes is a precursor to that described in Bhamidipati over two decades later. In fact, Bhamidipati discusses the Hayes-type pipeline, including its deficiencies, in the

background of the Bhamidipati invention.⁴⁰ Bhamidipati then offers an improvement to the Hayes pipeline that allows pipeline stages to be decoupled to ensure more optimal performance.⁴¹ The Examiner never-the-less cites Hayes for teaching that a fetch stage may be directly coupled to an execution stage.⁴² It is submitted that one skilled in the art would not be motivated to modify Bhamidipati with any teaching from Hayes, particularly when Bhamidipati is expressly providing an improvement to the Hayes pipeline.

The Examiner states that one skilled in the art would be motivated to make the cited combination because it would result in implementation of a standard, well-known, and easy to implement pipeline.⁴³ While the Hayes system is no doubt well-known 1970s technology, the Bhamidipati system provides an improvement targeted for today's high-speed data processing systems. Those skilled in the art of modern data processing systems are rarely motivated to turn the clock back more than two decades just because the earlier technology is easier to implement. Because there is absolutely no suggestion or motivation, either in the references themselves, or in the knowledge available to one skilled in the art, to modify Bhamidipati with Hayes, the Examiner has failed to set forth a prima facie case of obvious.⁴⁴

Further to the foregoing point, in expressly discussing a Hayes-type system, then providing a solution to the Hayes deficiencies, Bhamidipati is

⁴⁰ Bhamidipati column 1 lines 12-25 discuss.

⁴¹ Bhamidipati column 1 lines 24-35.

⁴² Paper 12 page 6 last paragraph.

⁴³ Paper 12 page 5, lines 5-8.

⁴⁴ *In re Oetiker* 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992).

teaching away from the direct coupling of stages used in the Hayes system and Applicants' claimed invention. Teaching away has long been considered an indication that the cited combination of references is improper.⁴⁵

Not only does Bhamidipati teach away from the cited combination of references and from Applicants' claimed invention, but Hayes does as well. As discussed above, Hayes teaches a system wherein instructions advance in lock step between all stages of the pipeline, including between the fetch and execution stages. While Hayes briefly notes the problems associated with this lock-step mechanism that are caused by varying fetch and execution times, the only solution Hayes offers involves allowing some instructions to by-pass some of the pipeline stages.⁴⁶ Thus, although Hayes recognizes some of the problems solved by Applicants' invention, Hayes actually *teaches away* from Applicants' claimed invention by suggesting that instructions should by-pass pipeline stages rather than being passed directly between those stages.

To summarize, there is absolutely no motivation to add any elements of Hayes to Bhamidipati. The Examiner has therefore failed to set forth a prima facie case of obviousness with respect to Claims 1, 21, 27, 32, 39, 40, and 46, and for this reason alone, these Claims should be allowed.⁴⁷ Moreover, even if elements of Hayes were added to Bhamidipati, it would not teach or suggest a system that allows fetch and execution stages of a pipeline to be both directly

⁴⁵ *In re Gurley*, 27 F.3d 551, 31 USPQ 2d 1130, 1131, (Fed. Cir. 1994).

⁴⁶ Hayes page 24, last paragraph.

⁴⁷ *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992).

coupled and independently operable. For this additional reason, these Claims are patentable over the cited combination of references.

II. Whether Claims 2-6, 22-25, 28-30, 33-38, 41-43, and 45 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes, and further in view of Kyker.

Applicants' Claims 2-6, 22-25, 28-30, 33-38, 41-43, and 45 each depends from an independent Claim considered in the foregoing section, and are patentable over the cited prior art for the reasons discussed above in the foregoing section. These Claims describe various other aspects of Applicants' invention that the Examiner asserts are taught by Kyker.

Kyker describes an improved branch recovery mechanism that is used to load correct micro operations into a pipeline after a branch misprediction has occurred.⁴⁸ Nothing in Kyker adds anything to Bhamidipati or Hayes to teach Applicants' invention that allows fetch and execution stages to be both directly coupled and independently operable. Applicants' Claims 2-6, 22-25, 28-30, 33-38, 41-43, and 45 are therefore patentable over Bhamidipati in view of Hayes, and further in view of Kyker.

III. Whether Claim 7 is unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes and Kyker, and further in view of Alferness.

⁴⁸ Kyker column 3 lines 48-63.

Applicants' Claim 7 depends from Claim 5, which is considered in Section II, above. Claim 7 is therefore patentable over the cited prior art for the reasons discussed above in the foregoing section. Claim 7 further describes the aspect of Applicants' invention wherein the execution circuit includes a microcode-controlled sequencer to control execution of extended-mode instructions. Extended-mode instructions are those instructions that require more than a single clock cycle to advance between successive execution stages of a pipeline.⁴⁹ The instruction set of Bhamidipati does not include these types of instructions. Instead, in Bhamidipati, it appears the execution stage X of any given instruction requires only a single clock cycle to complete.⁵⁰ Thus, one skilled in the art would not be motivated by anything in the references themselves or the prior art as a whole to add any type of microsequencer for controlling extended-cycle instructions to the design of Bhamidipati. Moreover, by citing four separate unrelated references, the Examiner is attempting to piece together the claimed invention in hindsight, something that has long been held impermissible.⁵¹ The Examiner has failed to set forth a prima facie case of obviousness with respect to Claim 7, and for this additional reason, Claim 7 is patentable over the cited combination of references⁵².

⁴⁹ Specification page 6 line 26 - page 7 line 7. See also, Alferness Figure 7, which shows extended-cycle instruction N requiring clock periods 1X - 4E to advance to the second execution stage.

⁵⁰ Bhamidipati column 2 lines 50-52, which discusses that each instruction takes four clock cycles to complete the four stages F, D, A, and X of the pipeline, and each clock cycle, the stages are operating on four different instructions.

⁵¹ *In re Gorman*, 933 F.2d 982, 986, 18 USPQ2d 1885, 1888 (Fed.Cir.1991).

⁵² *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992).

IV. Whether Claims 26, 31, and 44 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes and further in view of Alferness.

Applicants' Claims 26, 31, and 44 depend from various Claims discussed above in Sections I and II. These Claims are therefore patentable over the cited prior art for the reasons discussed above in these Sections. These Claims further describe the aspect wherein Applicants' execution stages include a microcode-controlled sequencer to control execution of extended-mode instructions.

As discussed previously in the foregoing Section III, it appears the instruction set of Bhamidipati does not include extended-mode instructions that require more than a clock cycle to advance between any two execution stages. Instead, from the Bhamidipati description, it seems all Bhamidipati instructions are "single-cycle" instructions that require a single clock cycle to complete any given stage of execution.⁵³ Thus, one skilled in the art would not be motivated by anything in the references themselves or the prior art as a whole to add any type of microsequencer for controlling extended-cycle instructions to the design of Bhamidipati. The Examiner has therefore failed to set forth a prima facie case of obviousness with respect to Claims 26, 31, and 44, and these Claims are patentable over the cited combination of references for this additional reason.

⁵³ Bhamidipati column 2 lines 50-52, which discusses that each instruction takes four clock cycles to complete the four stages F, D, A, and X of the pipeline. Although this discussion relates to Figure 1 designated as "prior art", the Bhamidipati system of Figure 3 is described as being incorporated into the pipeline of Figure 1 in column 3 lines 44-49.

Conclusion and Request for Relief

Applicants' Claims 1-7 and 21-46 are patentable over the cited combination of references. None of the references, alone or in combination, teach or suggest an instruction pipeline that allows instructions to advance within the fetch stages independently of the times at which they advance within the execution stages, while also allowing instructions to be provided directly by a fetch to an execution stage so that pipeline performance is optimized. Moreover, the Examiner has failed to set forth a prima facie case of obviousness because one skilled in the art would not be motivated to modify Bhamidipati with Hayes, particularly when Bhamidipati explicitly sets forth an improvement to the early Hayes-type design. The Examiner has failed to set forth a prima facie case of obviousness with respect to Claims 7, 26, 31 and 44 for the additional reason that one skilled in the art would not be motivated to add a microsequencer of the type described in Applicants' Claims to Bhamidipati, which includes an instruction set having single-cycle instructions. For all of these reasons, Claims 1-7 and 21-46 are patentable over the cited references. It is therefore respectfully requested that the rejection of the Claims be overturned, and the Claims be passed to issue.

Respectfully submitted,

Beth L. McMahon 07/16/04

Beth L. McMahon
Attorney for Applicants
Reg. No. 41,987
Telephone No. 651-635-7893
UNISYS Corporation
M.S. 4773
PO Box 64942
St. Paul, MN 55164-0942

Appendix A

Presented is a clean set of Claims 1-7 and 21-46 as last amended April 30, 2003.

Claim 1:

1 1. For use in an instruction processor that executes instructions included in a
2 predetermined instruction set at an execution rate determined by a system clock
3 signal, a synchronous instruction pipeline, comprising:
4 a pipeline execution circuit to process a first predetermined number of
5 instructions simultaneously, each of said first predetermined number of instructions
6 being in a respectively different stage of execution within said pipeline execution
7 circuit, instructions being capable of advancing to a next stage of execution within
8 said pipeline execution circuit at a time determined by the system clock signal; and
9 a pipeline fetch circuit coupled to provide each of the first predetermined
10 number of instructions directly to said pipeline execution circuit, the pipeline fetch
11 circuit to retain a second predetermined number of instructions simultaneously,
12 each of said second predetermined number of instructions being in a respectively
13 different stage of processing within said pipeline fetch circuit, an instruction being
14 capable of advancing to a next stage of execution within said pipeline fetch circuit at
15 a time determined by the system clock signal and independently of the times at
16 which instructions advance to a next stage of execution within said pipeline
17 execution circuit.

Claim 2:

- 1 2. The synchronous instruction pipeline of Claim 1, wherein said pipeline fetch
2 circuit includes an instruction queue to store a predetermined maximum number of
3 the instructions that are each ready to be processed by said pipeline fetch circuit.

Claim 3:

- 1 3. The synchronous instruction pipeline of Claim 1, wherein said pipeline fetch
2 circuit includes a pre-decode logic circuit to generate pre-decode signals for an
3 instruction that is in a pre-decode stage of processing within said pipeline fetch
4 circuit, and wherein an instruction can enter said pre-decode stage of processing
5 independently of the movement of instructions through said pipeline execution
6 circuit.

Claim 4:

- 1 4. The synchronous instruction pipeline of Claim 3, wherein said pipeline fetch
2 circuit includes a decode logic circuit coupled to said pre-decode logic circuit to
3 generate decode signals for an instruction that is in a decode stage of processing
4 within said pipeline fetch circuit, and wherein an instruction can enter said decode
5 stage of processing from said pre-decode stage of processing independently of the
6 movement of instructions through said pipeline execution circuit.

Claim 5:

5. The synchronous instruction pipeline of Claim 4, wherein said pipeline fetch circuit includes a first selection circuit coupled to said pre-decode logic circuit to allow an instruction to be received by said pre-decode logic circuit at a time determined by the system clock signal if said decode logic circuit is available to accept an instruction currently being executed by said pre-decode logic circuit.

Claim 6:

1 6. The synchronous instruction pipeline of Claim 5, wherein said pipeline
2 fetch circuit includes a second selection circuit coupled to said decode logic
3 circuit to allow an instruction to enter said decode stage of execution at a time
4 determined by the system clock signal if said decode logic circuit is not
5 processing another instruction.

Claim 7:

1 7. The synchronous instruction pipeline of Claim 5, wherein said pipeline
2 execution circuit includes a microcode-controlled sequencer to control execution
3 of extended stages of execution of extended-mode ones of the instructions,
4 wherein during said extended stages of execution, ones of the instructions being
5 executed by said pipeline execution circuit are not advancing to a next stage of
6 execution within said pipeline execution circuit, and wherein said first selection
7 circuit includes a control circuit to allow an instruction to enter said pre-decode

- 1 stage of processing while said extended-mode ones of the instructions are not
- 2 advancing to a next stage of execution within said pipeline execution circuit.

Claim 21

- 1 21. For use in an instruction processor, a synchronous pipeline circuit,
- 2 comprising:
 - 3 an execution circuit to provide a first predetermined number of execution
 - 4 stages, each being capable of performing a respective processing operation on
 - 5 a respective instruction; and
 - 6 a fetch circuit coupled to the execution circuit to provide a second
 - 7 predetermined number of fetch stages, each fetch stage being capable of
 - 8 performing a respective pre-execution operation on a respective instruction, the
 - 9 fetch circuit to transfer each instruction processed by the fetch circuit directly
 - 10 from one of the fetch stages to one of the execution stages, ones of the
 - 11 instructions processed within the fetch stages being capable of advancing to
 - 12 different available fetch stages independently of whether instructions are
 - 13 advancing within the execution stages.

Claim 22

- 1 22. The pipeline circuit of Claim 21, wherein one of the fetch stages includes
- 2 instruction address generate logic to predict which instructions are to enter the
- 3 fetch stages.

Claim 23

- 1 23. The pipeline circuit of Claim 22, wherein the instruction address generate
- 2 logic includes a circuit to clear ones of the fetch stages in response to a
- 3 determination that instruction execution was re-directed.

Claim 24

- 1 24. The pipeline circuit of Claim 21, and further including
- 2 a memory to store instructions;
- 3 a queue coupled to the memory to temporarily store at least one
- 4 instruction fetched from the memory; and
- 5 a circuit coupled to the queue and to at least one of the fetch stages to
- 6 fetch an instruction from the queue for presentation to at least one of the fetch
- 7 stages.

Claim 25

- 1 25. The pipeline circuit of Claim 24, wherein the circuit coupled to the queue
- 2 is capable of retrieving instructions from the queue for presentation to the at
- 3 least one of the fetch stages regardless of whether instructions are advancing
- 4 within the execution stages.

Claim 26

- 1 26. The pipeline circuit of Claim 21, wherein at least one of the execution
- 2 stages includes a microcode-controlled sequencer to control execution of

3 extended-mode instructions, and wherein during some stages of execution of the
4 extended-mode instructions, instructions are not advancing within the execution
5 circuit.

Claim 27

1 27. A synchronous instruction pipeline circuit for processing instructions
2 within a data processing system, comprising:
3 a first predetermined number of fetch stages to simultaneously process at
4 least a first predetermined number of instructions;
5 a second predetermined number of execution stages to simultaneous
6 process a second predetermined number of instructions, each received directly
7 from one of the fetch stages; and
8 wherein at least one of the fetch stages is capable of providing an
9 instruction to a different one of the fetch stages that is ready to receive an
10 instruction irrespective of movement of instructions between the execution
11 stages.

Claim 28

1 28. The pipeline circuit of Claim 27, wherein one of the fetch stages includes
2 address generate logic to predict which instructions are to enter the fetch stages
3 for processing.

Claim 29

- 1 29. The pipeline circuit of Claim 28, wherein the address generate logic
2 includes a circuit to flush one or more instructions from the fetch stages if it is
3 determined that a misprediction occurred.

Claim 30

- 1 30. The pipeline circuit of Claim 27, and further including:
2 a memory; and
3 a storage device coupled to one of the fetch stages and to the memory to
4 store instructions retrieved from the memory, wherein a predetermined number
5 of instructions may be stored within the storage device regardless of whether
6 instructions are advancing within the fetch stages.

Claim 31

- 1 31. The pipeline circuit of Claim 27, wherein one of the execution stages
2 includes a microcode sequencer to execute predetermined ones of the
3 instructions in a manner that may temporarily affect movement of instructions
4 within the execution stages.

Claim 32

- 1 32. A synchronous instruction pipeline to execute instructions, comprising:

2 an execution circuit having a first predetermined number of execution
3 stages to execute a first predetermined number of instructions substantially
4 simultaneously; and
5 a fetch circuit having a second predetermined number of fetch stages to
6 perform pre-execution operations on at least a second predetermined number of
7 instructions substantially simultaneously, one of the fetch stages being coupled
8 to provide each instruction processed by the fetch circuit directly to one of the
9 execution stages, at least one of the at least second predetermined number of
10 instructions being capable of advancing between different ones of the fetch
11 stages regardless of whether an instruction is being transferred by the fetch
12 circuit to the execution circuit.

Claim 33

1 33. The pipeline of Claim 32, wherein the fetch circuit includes an instruction
2 address generate section to determine which instructions are to enter the fetch
3 circuit.

Claim 34

1 34. The pipeline of Claim 33, wherein the instruction address generate
2 section includes a circuit to remove instructions from the fetch circuit during a
3 pipeline flush operation.

Claim 35

- 1 35. The pipeline of Claim 32, and further including:
2 a memory to store instructions; and
3 a queue coupled to store instructions from the memory, the queue further
4 being coupled to provide an instruction to the fetch circuit if one of the fetch
5 stages is available and irrespective of whether an instruction is being provided
6 from the fetch circuit to the execution circuit.

Claim 36

- 1 36. The pipeline of Claim 35, and further including a circuit coupled to the
2 queue to allow an instruction to be stored to the queue independently of whether
3 an instruction is advancing within the fetch circuit.

Claim 37

- 1 37. The pipeline of Claim 36, wherein the circuit allows a predetermined
2 maximum number of instructions to be stored to the queue independently of
3 whether an instruction is advancing within the fetch circuit.

Claim 38

- 1 38. The pipeline of Claim 37, wherein the one of the fetch stages includes a
2 circuit to allow retrieval of an instruction from either the memory or from the
3 queue.

Claim 39

- 1 39. The pipeline of Claim 32, wherein the fetch circuit includes a circuit that
2 allows instructions to advance within the second predetermined number of fetch
3 stages if one of the execution stages is performing a predetermined function.

Claim 40

- 1 40. A method of processing instructions within a synchronous pipeline of an
2 instruction processor, comprising:
3 a.) performing pre-execution operations on a first predetermined number
4 of instructions substantially simultaneously within a first predetermined number
5 of fetch stages of the pipeline;
6 b.) executing a second predetermined number of instructions
7 substantially simultaneously within a second predetermined number of execution
8 stages of the pipeline, wherein each of the second predetermined number of
9 instructions were received directly from one of the fetch stages; and
10 c.) allowing one or more of the first predetermined number of instructions
11 to advance between ones of the fetch stages independently of whether any of
12 the second predetermined number of instructions are advancing between ones
13 of the execution stages.

Claim 41

- 1 41. The method of Claim 40, and further including:
2 fetching an instruction from a memory;

3 storing the instruction within a queue; and
4 retrieving the instruction from the queue to undergo a pre-execution
5 operation within a predetermined one of the fetch stages.

Claim 42

1 42. The method of Claim 41, wherein at least one of the storing and the
2 retrieving step is performed independently of whether instructions are advancing
3 between ones of the execution stages.

Claim 43

1 43. The method of Claim 42, wherein ones of the steps are repeated for
2 multiple instructions.

Claim 44

1 44. The method of Claim 40, wherein one of the execution stages includes a
2 microcode-controlled sequencer for executing extended-mode instructions, and
3 further including executing one of the extended-mode instructions in a manner
4 that temporarily delays the advancing of instructions between ones of the
5 execution stages.

Claim 45

1 45. The method of Claim 40, and further including:
2 providing an indication that one or more predetermined operations are
3 occurring within one or more of the execution stages; and
4 in response to the indication, allowing instructions to advance within the fetch
5 stages.

Claim 46

1 46. A pipeline circuit for use in an instruction processor, comprising:
2 instruction fetch means for performing pre-execution operations on a first
3 predetermined number of instructions substantially simultaneously within a first
4 predetermined number of fetch stages;
5 instruction execution means for executing a second predetermined number of
6 instructions substantially simultaneously within a second predetermined number of
7 execution stages, each of the second predetermined number of instructions being
8 received directly from one of the fetch stages; and wherein
9 the instruction fetch means includes means for allowing at least one of the
10 first predetermined number of instructions to advance within the fetch stages
11 irrespective of whether instructions are advancing within the execution stage.